Diseño de interfaces de visualización para resultados de simulación

MARÍA VIRGINIA CIFUENTES[1], LUIS PABLO THOMAS[1], BEATRIZ M. MARINO[1]
{cifuente, lthomas, bmarino}@exa.unicen.edu.ar
[1] Instituto de Física de Arroyo Seco, Facultad de Ciencias Exactas, UNCPBA
Pinto 399, 7000 Tandil, Argentina
Teléfono (02293) 447110 Fax: (02293) 444433.

Resumen

Se presenta un modelo de diseño orientado a objetos que facilita la aplicación de estrategias de desarrollo concretas, algoritmos e implementaciones para el uso flexible de componentes genéricas en la visualización de resultados de simulaciones numéricas. De esta forma se reduce la complejidad de la visualización dinámica y simultánea de varias vistas necesaria en este tipo de interfaces, facilitando su implementación, extensión y reuso del diseño. El diseño se aplica, en particular, a la visualización de resultados de simulaciones de corrientes de gravedad sobre lechos permeables.

Palabras claves: patrones de diseño, microarquitecturas OO, simulaciones numéricas, agentes de control, corrientes de gravedad.

I. Introducción

Los modelos físico-matemáticos creados para describir diversas situaciones presentes en nuestro entorno son creados analizando las observaciones de una manera formal. Algunos modelos simplificados pueden ser resueltos analíticamente, pero la mayoría de los fenómenos complejos necesitan códigos numéricos para su solución. Los resultados de estos códigos consisten en un gran número de datos que deben representarse en diversas formas empleando variados recursos de visualización. Las conclusiones y predicciones así obtenidas pueden compararse cualitativa y cuantitativamente con las provenientes de los eventos reales, reflejando las correlaciones y las leyes subyacentes del fenómeno.

La programación del software de visualización frecuentemente adquiere una complejidad tal que hace necesario una programación basada en conceptos teóricos y prácticos avanzados a fin de lograr una adecuada modularidad, reusabilidad y extensibilidad del software desarrollado. De esta forma se encapsulan los detalles de implementación volátiles dentro de interfaces estables, reduciendo el esfuerzo requerido para entender y mantener el software, y facilitando su adaptación a nuevas aplicaciones. Este procedimiento se refleja también en una mejora de la ejecución, confiabilidad e interoperatividad de los programas desarrollados [Fayad+97].

En Cifuentes et al. (2001) se ha reportado un simulador de corrientes de gravedad (o densidad) desarrolladas sobre lechos permeables, las cuales aparecen frecuentemente en diversos fenómenos naturales. Allí se describe con detalle la formulación del problema físico-matemático y su resolución numérica por medio de un esquema explícito en diferencia finita de Lax Wendroff. Los resultados de la simulación completa se guardan en archivos que constituyen los datos usados en la herramienta de interface gráfica. También se incluye una interface de visualización simple de varios parámetros físicos relevantes de una o de varias ejecuciones diferentes en forma simultánea, escogiendo libremente el tamaño de las ventanas y las escalas de los gráficos de acuerdo con la importancia relativa asignada por el usuario, como así también la animación simultánea de los gráficos presentes en la pantalla. De esta forma se facilita el análisis de los datos obtenidos y su comparación cuando diferentes condiciones físicas son establecidas. Sin embargo, la interface desarrollada dificultaba la inclusión de nuevas representaciones y generaba problemas en la dinámica de animación al trabajar con varios archivos de datos, además de incrementar excesivamente la complejidad del software.

Se presenta aquí la herramienta de interface gráfica completamente rediseñada, lo que aumenta el número posible de parámetros representados y simplifica la animación. Sin embargo, el principal aporte de este trabajo reside en el marco conceptual y teórico del diseño del software, basado en los patrones que especifican y simplifican la estructura y el comportamiento del sistema intensivo de usuario. Las dificultades inherentes al tipo de datos usados y las representaciones necesarias, el uso de varias ventanas simultáneas y la animación de las vistas fueron entonces superadas con éxito. Los objetos son organizados por medio del patrón *Model View Controller* (MVC), donde se separan los resultados de la simulación (el modelo) de sus representaciones (las vistas), las cuales son preparadas y sincronizadas por medio de agentes (*controller*) [Gamma+95]. El patrón es implementado sobre la base del *framework Microsoft Fundation Class* (MFC), que es un *framework* de interface de usuario gráfica (GUI) muy utilizado para crear aplicaciones gráficas sobre una plataforma PC. Si bien MFC tiene limitaciones debidas a que no es portable para otro tipo de plataformas, su amplia adopción demuestra los beneficios para desarrollar aplicaciones gráficas [Fayad+97]. La animación de las vistas se implementó utilizando el patrón de diseño denominado *Observer*, dispuesto para que trabaje en forma cooperativa con el MVC. De esta forma se aprovechó el uso de patrones [Pre95], *frameworks* [Johnson88], librerías de clases y componentes para lograr una interface fácil de entender y mantener, además de perfomance y calidad interesantes.

En la próxima Sección se presenta el diseño general del software y luego se describe la herramienta de interface en particular. Si bien la interface se desarrolló para un problema específico, la organización modular basada en patrones y agentes le confiere una mayor generalidad, y el software puede convertirse fácilmente para visualizar los resultados de otros tipos de modelos. Finalmente, la última sección se dedica a las conclusiones.

II. Diseño de la interface de usuario

a) Marco Teórico y patrones de diseño empleados

a.1) Patrón *Model View Controller*

Como se mencionara previamente, se ha empleado el patrón *Model View Controller* (MVC), donde se organizan los objetos correspondientes a los resultados de la simulación (el modelo), sus representaciones (las vistas), y los agentes (*controller*). En forma gráfica,

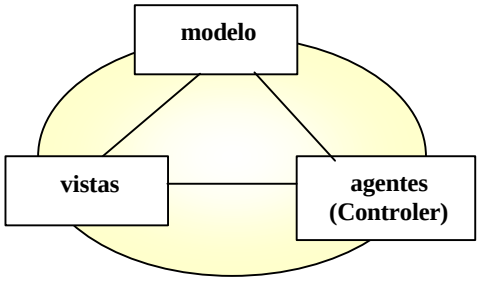

Figura 1 - Patrón básico de diseño MVC

Cada uno de los tres componentes principales de la interface se compone a su vez de elementos individuales, los cuales deben ser relacionados apropiadamente como se describe más abajo.

a.2) Patrón Observer

Las vistas son animadas en forma simultánea y coordinada, por lo que es conveniente que todas ellas estén asociadas a un solo archivo de datos. Para ello se empleó el patrón de diseño *Observer* clasificado como un patrón de comportamiento de objetos [Gamma+95] que define la dependencia *uno-a-muchos* entre objetos. El objeto sujeto a cambio se denomina *Subject* y los objetos dependientes se llaman *Observers*. La cantidad de *Observers* asociados a un *Subject* no está restringida. Cuando se produce el cambio de estado de un *Subject*, todos los *Observers* son notificados y actualizados automáticamente. La figura 2 muestra el diagrama de las clases empleadas y sus asociaciones dentro del patrón siguiendo los lineamientos de la notación UML (*Unified Modeling Language*) definida en [Booch+98].

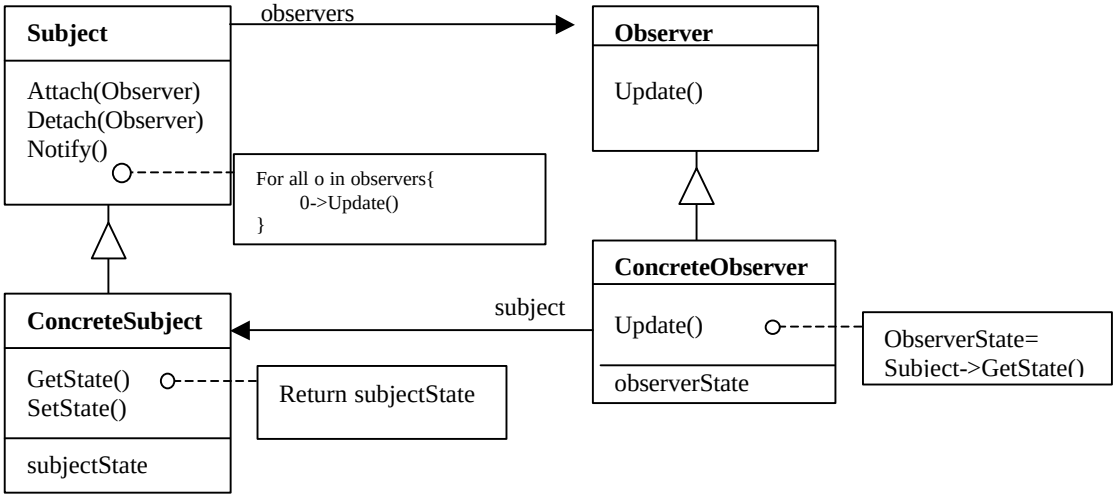

Figura 2 - Diagrama de clases en el patrón de diseño Observer.

Siempre que se produce algún cambio en un *ConcreteSubject*, éste notifica a sus *Observers* para evitar inconsistencias. Posteriormente, un objeto *ConcreteObserver* puede preguntarle al *Subject* para informarse y usa esta información para reconciliar su estado con el del *Subject*. La figura 3 ilustra las colaboraciones entre un *Subject* y dos *Observers*.

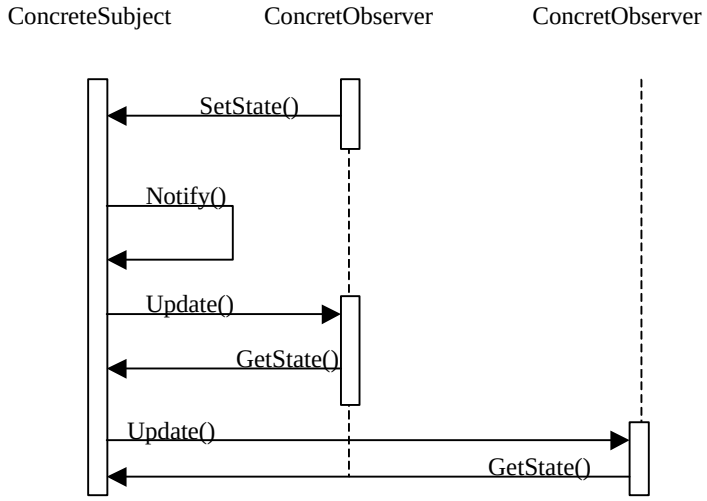

Figura 3 – Colaboración entre un subject y dos observers.

b) Desarrollo para el caso de visualización de resultados numéricos

Las clases que implementan las vistas y los agentes son responsables de la creación de la representación visual y de las entradas provocadas por el usuario vía teclado y/o *mouse*. Cada agente selecciona un subconjunto de datos presentes en los resultados (modelo), que generan una vista representativa de la simulación numérica.

En el caso particular de resultados de simulación, el modelo normalmente está compuesto por un conjunto de valores de diferentes parámetros físicos que dependen del tiempo t en la evolución del fenómeno, y de su posición en un cierto sistema de coordenadas. Estos parámetros pueden ser representados en tres formas básicas diferentes ilustradas en la figura 4. Para las vistas de las *plantillas de datos*, los agentes generan vistas tabulares de los datos del dominio. Otro tipo de vista surge de la representación de valores de un parámetro en un dado tiempo por medio de agentes que podrían denominarse *espaciales*. Los datos así representados constituyen una fracción pequeña A del conjunto de resultados del modelo correspondientes al tiempo *t*. Los agentes *temporales*, por otra parte, emplean resultados generados durante un lapso grande de la evolución del fenómeno, que puede ser la totalidad de la evolución del sistema físico (subconjunto B en la figura 4).

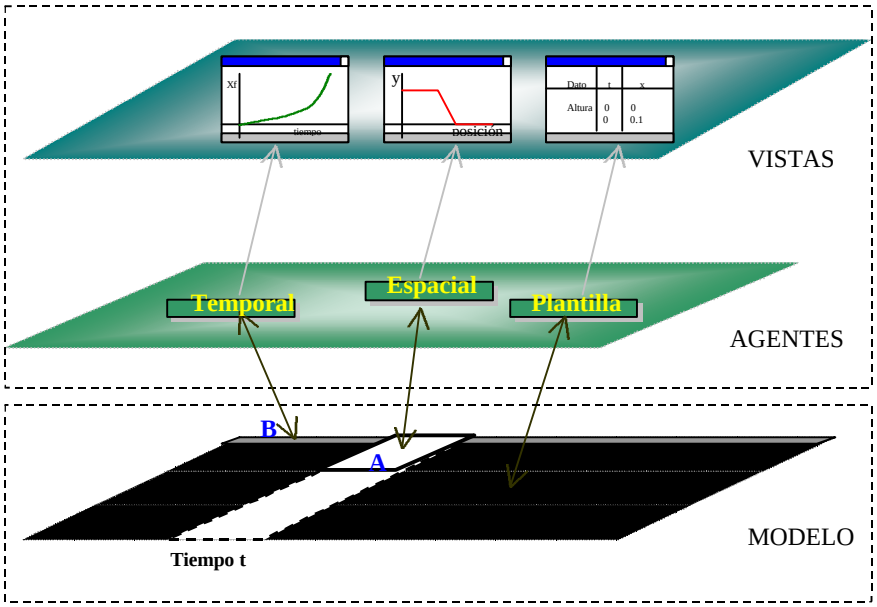

Figura 4- Componentes de la interface del usuario.

Varias vistas activadas simultáneamente permiten observar diversos parámetros de interés en el mismo tiempo físico. La animación en el caso de agentes espaciales es realizada por medio de una representación secuencial de fracciones próximas del conjunto de resultados. Para los agentes temporales, los resultados en el instante de la animación ($A \cap B$) son señalados por medio de indicadores especiales en las vistas (se aclara en la próxima Sección).

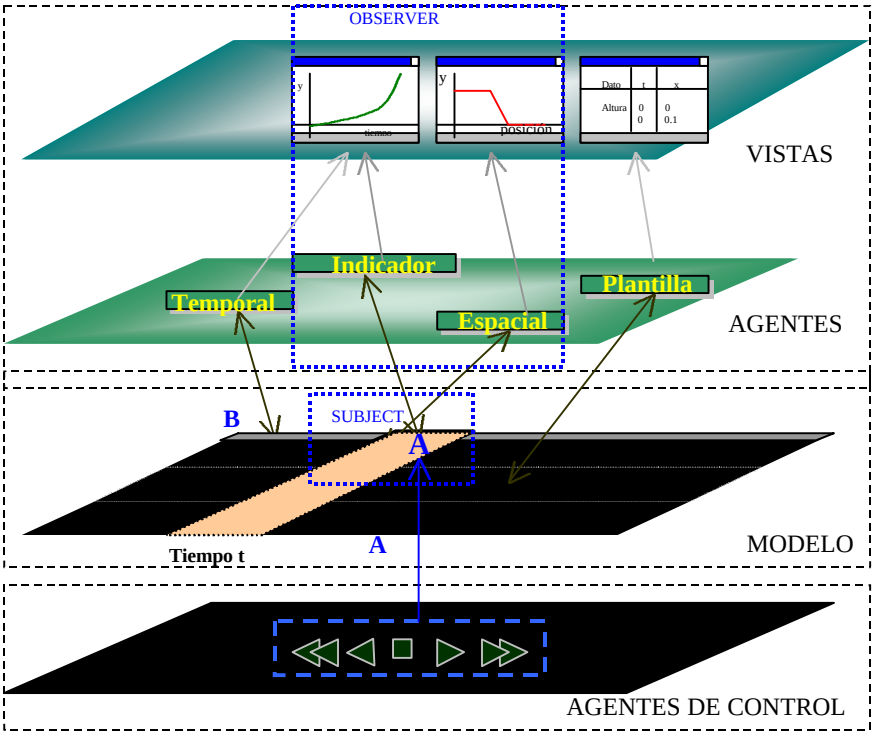

Figura 5 – Identificación del *Subject* y de los *Observers*.

La dinámica de actualización está correlacionada con la secuencia temporal de los datos en diversas formas (Avance, Avance rápido, retroceso, retroceso rápido, etc.) que modifican el *Subject* del patrón *Observer*. Los resultados del código numérico son ubicados en un único archivo para que resulte fácil definir el *Subject*. La figura 5 muestra un diagrama en el cual se pueden observar el *Subject* (conjunto A de datos) y los *Observers*.

Una vez definidas las vistas activadas en el área de cliente, el usuario interactúa con el modelo a través de los *agentes de control*. Se aplican a las *vistas espaciales* y a los indicadores de las *vistas temporales*, mientras que las plantillas y la estructura de las vistas temporales quedan intactas. Los distintos *agentes de control*: *retroceso, retroceso rápido, avance, avance rápido, y detener* se refieren a la elección secuencial de distintos subconjuntos temporales para la representación de las vistas, así también como la velocidad con que son actualizadas. Dada la diversidad de parámetros que pueden representarse, el análisis de las distintas vistas ayuda al usuario a captar fácilmente la evolución subyacente en el modelo, estudiando y comprendiendo lo que sucede en cada estadio temporal de un fenómeno. Al observar simultáneamente la evolución de varios parámetros de interés, las animaciones tornan más ventajosas a las vistas.

III. Aplicación a la simulación de corrientes de gravedad

Apliquemos ahora la arquitectura mencionada a los resultados provenientes de la simulación de corrientes de gravedad sobre lechos permeables [Cifuentes+01]. Las corrientes de gravedad son flujos generados por una diferencia de densidad entre el fluido que compone la corriente y el del medio ambiente que la rodea. Los fluidos son incompresibles, miscibles y en ciertas circunstancias la coordenada transversal puede ser ignorada. La configuración estudiada consiste en un *fluido*

ambiente, de densidad $\tilde{n}_1$, que se halla en reposo dentro de un canal de sección rectangular horizontal (ver figura 6). El fluido de densidad $\tilde{n}_2 > \tilde{n}_1$ se halla inicialmente en reposo dentro de una caja de altura h_0 y longitud x_0. La corriente de gravedad se genera cuando el fluido con mayor densidad es liberado dentro del fluido ambiente desplazándose a lo largo del eje *x*, bajo la acción de la gravedad *g* que actúa en dirección vertical [Bonnecaze+93] [Buffard+98].

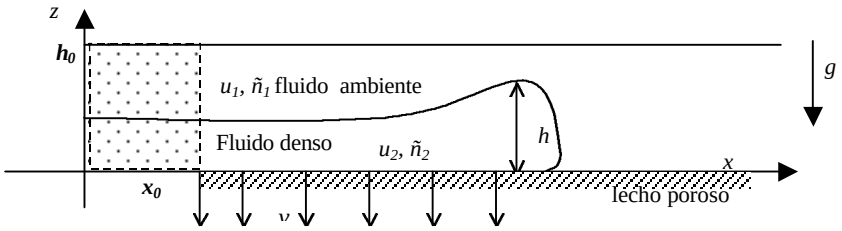

Figura 6 - Esquema del modelo físico.

Los resultados de la simulación se organizan en un archivo de datos compuesto de columnas correspondientes al tiempo *t*, la posición del frente *x*, la altura de la corriente *h(x,t)*, la profundidad alcanzada en el medio poroso o *drenaje d(x,t)*, la componente horizontal promedio de la velocidad $u_2(x,t)$ y la componente vertical de la velocidad *v(x,t)*. El archivo que contiene el resultado de una simulación constituye el *modelo* en la arquitectura mencionada anteriormente. La elección de un intervalo de *tiempo* específico t_1 conforma el *subconjunto* A*(t)* donde están los datos de la altura *h*, del drenaje *d* y de las velocidades u_2 y *v* para todas las posiciones en el tiempo t_1.

Además, la aplicación desarrollada manipula estos archivos como documentos e implementa una interface de documentos múltiples (MDI). Esta interface le permite al usuario trabajar con diversos archivos simultáneamente brindándole, por ejemplo, la ventaja de comparar resultados provenientes de distintas ejecuciones del modelo donde se han empleado valores dispares en los parámetros.

Los *agentes Plantillas* permiten visualizar los resultados de la simulación en forma de tablas de datos. Por su parte, los *agentes espaciales* son los responsables de crear las vistas de los *perfiles de altura*, en los que la altura es representada en función de la posición *h=h(x)* para el tiempo escogido, como se ilustra en la Fig. 6. En forma análoga se generan las vistas de los *perfiles de drenaje d=d(x)* y las vistas de los *perfiles de la velocidad horizontal* $u_2 = u_2(x)$ y *vertical v=v(x)*.

Los *agentes temporales* son responsables de crear las *vistas temporales* en donde se representan los valores del parámetro escogido en función del tiempo. El parámetro escogido debe tener un solo valor para cada instante y el usuario visualiza entonces su evolución temporal en un lapso prefijado menor o igual que el tiempo total de la simulación numérica. Las *vistas temporales* generadas en esta aplicación corresponden a la *masa sobre y debajo del medio poroso*, la *posición* x_f *del punto mas avanzado o frente*, la *altura máxima* h_{max}, y la *velocidad del frente* dx_f/dt, todos ellos referidos al fluido denso.

El dominio representado en las *vistas temporales* surge de aplicar funciones específicas a los resultados. Por ejemplo, si se integran los perfiles de altura *h* y los perfiles de drenaje *d*, se halla el dominio de la *vista de la masa sobre y/o debajo del medio poroso*, respectivamente. El dominio de la *vista de la posición del frente de la corriente* se calcula hallando el valor máximo de la abscisa para la cual *h>0*. El archivo así generado constituye el *subconjunto* B mencionado en la figura 4, que se integra al *modelo.*

Por último, los *agentes de control* que son responsables de la dinámica de actualización de las vistas (avance, avance rápido, retroceso, retroceso rápido, detener y animar) varían según se

apliquen a las *vistas espaciales* o a las *vistas temporales* como se mencionó anteriormente. En la figura 7 se ilustra el *agente de avance* aplicado a varias *vistas espaciales* de una simulación sobre un piso permeable horizontal. Los perfiles activados en el área de cliente corresponden a los perfiles de altura y drenaje (h/h_0), la velocidad horizontal sobre el lecho poroso (u/u_0) y la velocidad (v/v_0) de penetración en el lecho poroso, adimensionalizados apropiadamente [Cifuentes+01]. Cuando se aplica el *agente de avance* se produce la animación simultánea de los perfiles.

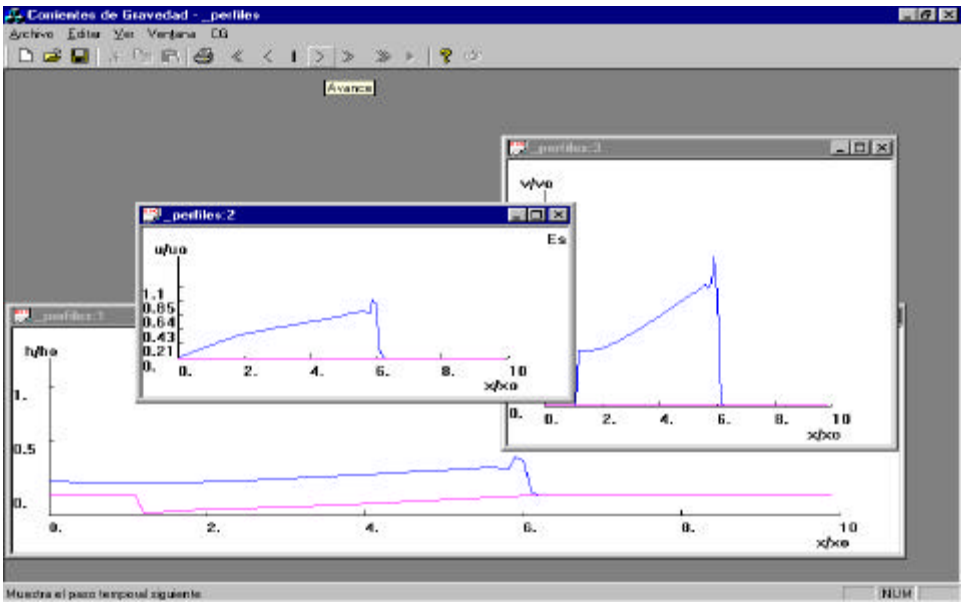

Figura 7-Vista del área cliente con las *vistas de* perfiles de altura (abajo, azul), *perfil de drenaje* (abajo, rosa), com*ponente horizontal de la velocidad* (arriba, izquierda, azul), *componente vertical de la velocidad* (arriba, derecha, azul) en una disposición irregular correspondientes a un modelo de simulación de piso permeable. La animación realizada por los *agentes de control* se verifica en la gráfica de los distintos perfiles.

En la figura 8 se observan las vistas *espaciales y temporales* activadas en forma simultánea. La animación realizada por los *agentes de control* se verifica en el desplazamiento horizontal de la línea roja (es decir, el *indicador* mencionado anteriormente) en las *vistas temporales* y en la superposición de nuevos perfiles (en las *vistas espaciales*).

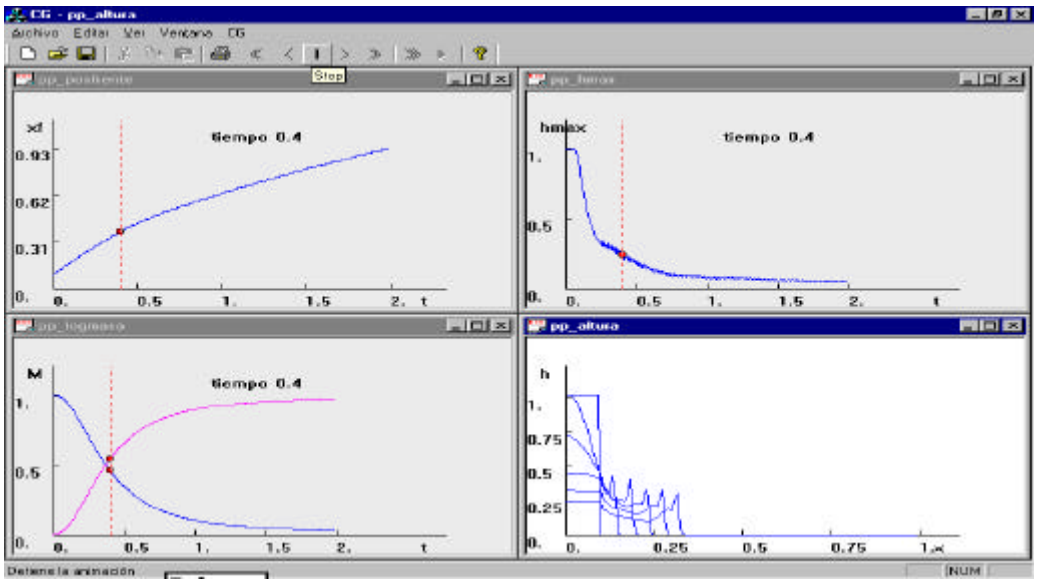

Figura 8-Vista del área cliente con las *vistas temporales* de posición del frente (arriba, izquierda), variación de la altura máxima (arriba, derecha), variación de la masa (abajo, izquierda) y la *vista espacial del perfil* de altura (abajo, derecha) en una disposición en mosaico correspondientes a un modelo de simulación de piso permeable.

IV. Conclusiones

Este trabajo describe el diseño de una interface de visualización dinámica para analizar los resultados de simulaciones numéricas basado en la aplicación de los patrones *Model View Controller* y *Observer*, el uso de agentes y de *frameworks* de aplicación OO (MFC) junto a librerías de clases y a componentes genéricos. El diseño reduce la complejidad de la visualización dinámica y simultánea de varias vistas convenientes para este tipo de interfaces, facilitando su implementación, extensión y aplicación a otros resultados. De esta forma se mejora la calidad del *software* con un menor trabajo de programación. La visualización dinámica posibilita la comparación de parámetros de la misma simulación, o bien la comparación de distintos parámetros en distintas simulaciones. El diseño puede aplicarse tanto a resultados de códigos de simulación unidimensionales como bidimensionales con cambios menores.

Este trabajo ha sido financiado con fondos provenientes de PIP-CONICET y UNCPBA.

Referencias

[Bonnecaze+93] Bonnecaze R T, Huppert H E and Lister J R (1993) Particle-driven gravity currents. J. Fluid Mech. 250: 339-369.

[Booch+98] Booch G, Rumbaugh J and Jacobson I (1998) The Unified Modeling Language User Guide, published by Addison-Wesley.

[Buffard+98] Buffard T, Gallouet T , Herard J M (1998) A naive scheme to solve shallow-water equations. Comptes Rendus de l' Academie des Sciences serie 1-Mathematique 326: 385-390.

[Cifuentes01] Cifuentes V, Thomas L and Marino B (2001) Simulador de corrientes de gravedad sobre lechos permeables, 30 JAIIO.

[Fayad+97] Fayad M and Hamu D (1997) Object-Oriented Enterprise Frameworks: make vs. Buy Decisid Guidelines for Selection, The Comunications of ACM.

[Gamma+95] Gamma E, Helm R, Johnson R and Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software, published by Addison-Wesley.

[Johnson88] Johnson R and Foote Brian (1988) Design Reusable Classes, Journal of Object-Oriented Programming, SIGS.,1,5,22-35.

[Meyer97] Meyer B (1997) Object-Oriented Software Construction. Second Edition, published by Prentice Hall.

[Pree95] Pree Wolfgang (1995) Design Patterns for Object-Oriented Software Development, published by Addison-Wesley.